Excerpts from

THE MISOSYS QUARTERLY

LB

LB86

Data Management

Little Brother

Little Brother (LB) is a data management system where ease of use is its primary goal. With LB, you don't need to program ANYTHING or remember complicated command sequences to manage your data. Even for the most complex data management needs, Little Brother will produce results very quickly, often with just a few keystrokes. This is because EVERY function in LB is menu driven and comes with complete on-line HELP information which is always at your fingertips.

With LB, you can concentrate on what you do best - managing your data, and leave the programming to us. We've put all of our design and programming expertise into LB so that your data management needs can be satisfied quickly and 'painlessly'. LB will handle almost any data base needs that you may have. Virtually the only limitation is your available disk space.

You easily define the layout of your data records. LB will handle up to 65534 records, and each record can contain up to 1024 characters. LB supports up to 64 different data fields for each record, where each field may be from 1 to 254 characters long. There are seven types of data fields available:

Alpha - Only the letters A-Z (a-z) and <space> may be entered.

Numeric - Only digits (0-9), a period and a minus sign may be entered.

Right Justified - Numeric with the value displayed and printed with 'leading spaces'.

Literal - Any ASCII character may be entered.

Dollar - 'dollar' values, with up to eight digits allowed to the left of the decimal point.

Float - 'floating point' values with 8 digits to the left and right of the decimal point.

Calculated - Allows calculations to be performed using any 'number' field (i.e. R, J, N, D, or F). The calculation is user defined, and may include addition, subtraction, multiplication and division. Calculations are precise up to sixteen significant digits.

Defining a LB data base is simple. Just enter a descriptive name for each field, the type of field that it is (e.g. "D" for Dollar, "L" for Literal, etc) and the length of the field. LB even has provisions for defining a "Protected Field", so that the data for that field will not be displayed unless the proper "Password" is entered. Full editing capabilities are available when defining a data base.

After defining your data layout, all you need to do is establish a "screen", and you are ready to begin entering data! Up to 10 different screens may be used to display your data. Having entered information, you may view or edit any record at any time. It is always a quick and easy operation to "Find" information with LB. You can even create an "Index" to your data by sorting the information in any non-calculated field, so that your data records can be accessed in either "ascending" or "descending" order. Using an Index will allow you to find any piece of information within a matter of seconds, even if there are tens of thousands of data records in your data base!

Once you have built a data base, you may wish to print the information. Simply define a print format, and LB will print the records according to your specified format. Up to 10 different print formats can be created. LB can handle almost any kind of print format; you can print listings complete with headers/footers, date, time, page numbering, totals and sub-totals if desired, mailing labels format, and even form letters. As with any data-related operation when using LB, you select what records get printed (according to your specified criteria). Records can be printed in "sorted" order as well, which is great for organizing your report (especially useful for "zip code" zoning).

Cont'd on inside rear cover

Product Highlights: Little Brother

A Little Brother user needed help in getting to first base with LB and in developing a screen definition. What was essentially needed was an outline of the steps neccessary to create a usable data base. What follows is such an outline."

There are approximately three processes that need to be accomplished in order to create a data management system using LB. The first is to define the data file environment. This task establishes the different items (data fields) which will appear in the data file. This part corresponds to your illustration of name, address, etc.

Once you have established the structure for a data file, two more things need to be accomplished. You have to provide a means for printing the information and you have to provide a means for entering information into the data file. You apparently understand the process of defining the output printing formats discussed on pages 19-58 of the LB manual.

The DEFINE SCREEN FORMAT section discusses that task associated with telling LB how you want to see the input screen for entering data. You see, other database programs which allow you no control over the input screen leave you no facility for customizing the input. With LB, you can easily create the input screen. All you do is follow the instructions on pages 7-18. You move the cursor around on the screen until it is positioned at the screen location where you want to see the request to enter a field then type in the appropriate prompt and field specifier. Using your example, if you just want to enter each item on a separate line, then assuming your example fields are numbered in the order presented in your letter, the following input screen would accomplish this:

```
Name: ^1^
Address: ^2^
City: ^3^
State: ^4^
ZIP code: ^5^
Phone: ^6^
```

Of course, you could dress up the screen with a heading such as, "Customer Address File". You certainly could position the input fields according to your taste - perhaps putting City, State, and ZIP on one line.

A useful reason for giving you the capability of defining your input screen would be to isolate data entry to a subset of the fields in your data file. You may have a large number of fields which do not get input but are calculated. Or maybe some fields are entered rarely. You can have different input screens to use at different times - each input screen tailored to the data which you want to enter. Having less fields on the input screen gives it a less cluttered look.

I am confident that you will find LB's methods easy to use. Don't forget that you can always change an input screen's presentation - even after you have begun using it to enter data.

Another user wanted to duplicate a Little Brother data base structure. Turns out the job is extremely easy IF you do a little preplanning. If you have any reason to suspect that you may want to duplicate your data base, save a copy of the DEF, LB, VDn, and PRn files BEFORE YOU ADD ANY RECORDS. Then all you need do to create subsequent data bases using the same data definition is to use backups of the "saved" set. Using a different set of "filenames" helps. If you have not preserved a copy of these original files, you will have to recreate the data file definition; however, the VDn and PRn files may be reused. The definition can be easily recreated by using a printout of the field definitions obtainable from the "define data field format" function using the existing data base.

Now speaking of existing data bases, I recently went through a revision of the customer file database here at MISOSYS. A little background is in order. Prior to the acquisition of LSI's retail operation, we used PowerMAIL to hold our base of registered customers (we only kept those interested enough in returning a

registration card). Logical Systems used PROFILE in the past and then Little Brother after they completed that package. We, of course, acquired their data base. Now LSI had about 50,000 records in that LB data base. About 12,000 were identified as recent purchasors or registered customers. The remaining 36,000 represented inquiries, subscription lists of defunct magazines, and other entries we didn't feel warranted our direct mailings. Thus we wanted to trim their list down to 12,000. We also wanted to drop some fields and add others (specifically fields relating to THE MISOSYS QUARTERLY subscriptions). That meant a significant use of the Little Brother Maintenance Utility. We also wanted to merge our data base into the revised LSI base. That meant a bulk loading using the AUTO option of Little Brother.

The LSI base runs on an old Leading Edge PC e/w a 10 Meg hard drive. Since the 50,000 record base used up 8.5 megs of the hard drive, we needed a larger hard disk to provide room to store both the old base and the new one. To do this, we acquired a 20Meg drive package with controller and cables for $399 - good deal. Turns out that it would not work in the Leading Edge machine but worked perfectly in the IBM PC we had. So much for compatibility of the older LE. So, using the PC for the job, we backed up the 50,000 record base onto about 25 floppies and restored the base to the IBM's 20Meg drive. We then defined a new data base structure for our new base using the LB Maintenance Utility (LBMU). We extracted the 12,000 "active" records and populated the new base with them - all this is part of the LBMU. We then backed up the new data base onto floppies and restored them to the LE machine after purging all of the old data base files.

The next job was to get our PowerMAIL data over to the new data base. Using the PMAIL's function to extract records into an adder file, we generated a 4,000 record adder file. Now the AUTO facility of LB works something like LDOS's redirection capability - it allows you to obtain the input, which normally would come from the keyboard, from a disk file. All I had to do now was create a disk file that would look like the keystrokes I would type to input 4,000 records. Easy stuff!

As part of their technical support literature, LSI had available an example of a conversion utility written in BASIC to take a fixed record length file and convert it to a LB AUTO-input file. Since it was not published in the last LSI JOURNAL, I think it appropriate to publish it here.

```
10 OPEN"R",1,"OLDDATA",LRL: 'LRL must match existing file
20 FIELD 1, ... ...: 'FIELD the old file to match its data structure
30 OPEN"O",2,"AUTO/JOB": ' OPEN the LB auto file
40 PRINT #2,"2";CHR$(13);"A";
50 FOR L=1 TO LOF(1): 'DO all records in old data file
55 GET 1,L
60 PRINT#2, field variable 1;CHR$(13);
70 PRINT#2, field variable 2;CHR$(13);
80 ETC... (DO ALL FIELDS IN OLD FILE)
90 PRINT #2,CHR$(27);
199 NEXT L
110 PRINT#2,CHR$(26);
120 CLOSE:END
```

The semicolons after the "PRINT #2" statements are VERY important - do not forget them! If there are blank records at the end of the old data file, line 50 should be changed to read: FOR L = 1 TO real number of records.

Now one more important thing to remember is that if your new Little Brother data base has additional fields or it no longer includes some of the fields which were in the old one, then take that into consideration. A carriage return (CHR$(13);) should be PRINT#'d in the proper position for each new field. And

don't PRINT# any field from the old data file which is not used in the new file.

As a more concrete example, here's a copy of the BASIC program I used to convert my PowerMAIL file to the auto input file of Little Brother. This program was written for use with our EnhComp BASIC compiler on the Model 4.

```
'MDATA/BAS - 06/09/86
ALLOCATE 2: OPEN "r",1,"pmail/add",128
FIELD 1, 15 AS F0$, 10 AS F1$, 20 AS F2$, 20 AS F3$, 10 AS F4$, 15 AS F5$, 8 AS F6$,
10 AS F7$, 5 AS F8$, 12 AS F9$, 1 AS FLG1$, 1 AS FLG2$, 1 AS FLG3$
OPEN "o",2,"mdata/dat:1"
PRINT#2,2:PRINT#2,"A";
FOR I = 3 TO LOF(1): GET 1,I: 'Skip the header sector in PMAIL/ADD
IF ASC(LEFT$(F0$,1)) = 255:' Do not bother with deleted records
    NEXT I
ENDIF
PRINT#2,!STRIP$(F2$):'Company name field
PRINT#2,!STRIP$(F0$):'Last name field
PRINT#2,!STRIP$(F1$):'First name field
PRINT#2,!STRIP$(F3$):'Address 1 field
PRINT#2,!STRIP$(F4$):'ADDRESS 2
PRINT#2,!STRIP$(F5$):'CITY - FIELD 6
IF ASC(FLG3$) AND 32:' if foreign customer
    PRINT#2,"":       ' do not print contents of STATE field
ELSE
    PRINT#2,!STRIP$(F6$)
ENDIF
PRINT#2,!STRIP$(F7$):'ZIP code field
IF ASC(FLG3$) AND 32: if foreign customer
    PRINT#2,!STRIP$(F6$):'print STATE field now (Country)
ELSE
    PRINT#2,"":'else country field blank for US
ENDIF
PRINT#2,"":'FIELD 10 - MAIL CODE
IF F8$ = "     ":'data1 field stored last purchase date
    PRINT#2,""
ELSE
    PRINT#2,LEFT$(F8$,2)+"/"+MID$(F8$,3,2)
ENDIF
PRINT#2,"86/06":' ORIGIN DATE
PRINT#2,CHR$(13);CHR$(13);CHR$(13);:' PRINT 3 <ENTER>S
IF ASC(FLG1$) AND 128:'Keep record of MC purchase
    O$="C"
ELSE
    O$=""
ENDIF
PRINT#2,O$
PRINT#2,CHR$(27);:' SAVE THE RECORD
NEXT I
PRINT#2,CHR$(26);:CLOSE:END
FUNCTION STRIP$(S$):'   my function to strip trailing spaces
IF LEFT$(S$,1) = " ":'  just used to make the output file smaller
    RETURN ""
ENDIF
LOOP%=LEN(S$):INC LOOP%
REPEAT
    DEC LOOP%
```

```
UNTIL MID$(S$,LOOP%,1) <> " "
RETURN LEFT$(S$,LOOP%)
```

There you go. It's as simple as that. Now there is no excuse for not bringing your old data files up to the flexibility of Little Brother data management. Oh, by the way. In order to get this processed file over to the PC from the MAX-80 where the old data was stored, I direct connected the MAX to the PC with a null modem and XMODEM'd the MDATA/DAT file. Now the file was about 400K so it took a while at 1200 baud; thet's why I added that STRIP function.

Converting MailFile to Little Brother

Here's a little exercise we did for a couple of reasons. First, it may show a few folks just how easy it is to define a Little Brother Data Base. Second, it provides a way to port your old MailFile data files over to Little Brother for either Model 4 use or MS-DOS use (with little or no work required).

Now certainly, it would make more sense to embellish the data fields to suit your own needs; something you couldn't do with MailFile. On the other hand, since I can't second guess your needs, you'll have to be content with an emulation of MailFile using Little Brother; or define your own record layout and then adapt the conversion program discussed later. What I have done is defined a data structure identical to MailFile; defined an add/edit/update screen which looks like MailFile, and defined two output reports: one-across mailing labels and an 80-column directory; the resulting set of LB data files can be considered a MailFile template. I also wrote a BASIC program to take an existing MailFile data base and convert it for bulk loading into the Little Brother emulation; the bulk load is a "job" file.

The five files which make up the MailFile template are on DISK NOTES 7. They are MAILFILE/DEF /LB, /PRO, /PR1, and /VDO. In addition, I have typed in a LB job file to automatically generate two index files. The first index sorts the data by name; the second sorts by ZIP code. Both sorts are just what you would experience by the constant sort of MailFile. The job file is named MFINDEX/JOB.

It's kind of tough to demonstrate the ease of defining a data base without at least telling you the steps which I went through. Thus, the following scenario outlines what I did. I'll include the menu screens as I work through the scenario so those readers who do not have Little Brother will be able to follow.

When you invoke LB, you will see the main menu. Here's what it will look like:

```
Little Brother - LSI Database Version 1.0.0
Copyright (C) 1985 by Logical Systems, Inc.

                1) Select Data Base Name

     2) Add Records                      8) Define Screen Formats
     3) Update or Delete Records         9) Define Print Formats
     4) Print Records                   10) Define File Format
     5) Sort or Select Records          11) Set Screen/Add Index
     6) Run Automatically               12) Change Password
     7) Expand Data File                13) View Field Definitions

Name: MAILFILE        Index:None    Screen:1   Allocated:100   Used:76

Enter Selection Number ..
```

This main menu screen actually shows the results of the entire job. That's why the bottom status line shows 76 records in use out of 100 allocated. I already loaded a MailFile database into this demo copy.

Now the first step we need to do is menu item 10 - define the data file record layout. This then prompts for the data base name (where I specified "MAILFILE") and the data base password which will be needed later to view definitions and make screen changes. I also entered "MAILFILE" as this password; that, then will be the name and password on the DISK NOTES 7 template.

The <D>efine command is used to initiate LB prompts for field information. The LB prompted entries are easy to specify; you just enter a field name, that field's type, that field's length, and whether it is protected or not from normal editing. Thus, specifying the nine MailFile fields gave me the following chart:

```
1  Name: Name #1      Type: L      Length:  27  Protect: N
2  Name: Name #2      Type: L      Length:  27  Protect: N
3  Name: Address      Type: L      Length:  27  Protect: N
4  Name: City         Type: L      Length:  15  Protect: N
5  Name: State        Type: U      Length:   3  Protect: N
6  Name: Zip Code     Type: L      Length:  11  Protect: N
7  Name: Phone        Type: N      Length:  14  Protect: N
8  Name: Ext          Type: N      Length:   4  Protect: N
9  Name: Code         Type: L      Length:   6  Protect: N
```

<RET> continues .

Note that the L, U, and N types stand for literal, upper case literal, and numeric. For those not in the know, other types supported by Little Brother are "A" alphabetic, "B" upper case alphabetic, "R" right justified, "D" dollar, "F" floating point, and "C" calculated (using add, subtract, divide, multiply).

Now that the data base has been defined, the next reasonable course of action is to define the screen format to be used for adding data, searching for data, and editing data. We exit the "define file format" module and select number 8, Define screen formats.

MailFile Emulation using Little Brother

```
1> Name #1    : ..........................
2> Name #2    : ..........................
3> Address    : ..........................
4> City       : ..............
5> State      : ...
6> Zip Code   : ...........
7> Phone      : ..............
8> Ext        : ....
9> Code       : ......

                        Define Screen 1
Help        Quit        Define      Edit        Print       Save
```

It's easy to layout a screen; all you do is move the cursor about the video screen using the ARROW keys and type in your information. When you want to designate a

data field, a simple CTRL-F will prompt for the "field" number. LB also allows you to switch on/off inverse video highlighting, delete field designations previously designated, insert/delete any characters which you typed, and insert/delete entire lines. I easily constructed the previous screen which can be used for adding records, viewing records, and editing records.

Now that the data base file has been defined and the input screen layed out, the next step is to design any needed output reports. Quitting the "define screen" option, I'm back at the LB main menu. I select option 9, "Define printer format".

Generating the structure of an output report is about as easy as laying out the input screen. The cursor is moved about the video screen and any characters which you want typed on the output are just typed into the screen. Fields from a data record which are to appear on the printout are designated by a sequence such as, "^4^", which designates field 4.

Since an output report can (and usually does) exceed the 80 columns of your video screen, LB provides a windowing technique to provide for wider reports. In fact, three separate windows are dislayable to give you up to 208 virtual columns for your report.

The twenty columns available for designating rows of information can also be divided into header, record data, and footer areas. I wanted a header which included the name of the data base, the current date and time the report was generated, and a page number. I also wanted a footer to print a separating line followed by a blank line. Thus, I divided the six lines of information which I typed into the video screen into two lines for a header, two lines for printing record data, and two lines for a footer. Here's the results of my 80-column "MailFile directory" report.

```
<MailFile 80-Column Directory> <File:^f^> <^d^ ^t^>   Page: ^p^
================================================================================
^r^   ^1^                    ^2^                              ^7^          ^8^
      ^3^                    ^4^            ^5^ ^6^           ^9^
================================================================================

..............
                          Print Screen - File # 1
Header : 1-2     Text : 3-4    Footer : 5-6    1 Rec: N  Row :1     Column:1

<V>iew, <E>dit, <F3> Save, <ESC> Quit - Enter Command:  .
```

The series of dots represents blank screen lines which I have omitted to avoid wasting QUARTERLY paper. Incidentally, these "screen images" were captured by routing the *PR device to a disk file before entering Little Brother and then invoking a DOS "screen print" command whenever I wanted to capture a screen. I subsequently loaded the "screen print" disk file into ALLWRITE for further editing (i.e. adding the scenario text which you are now reading). This technique of screen printing to a disk file is quite useful (now if we could add that feature to MS-DOS!).

I also wanted to create the "output report" for printing mailing labels. Duplicating the procedure used for the 80-column report, I worked up the following screen:

```
^2^
^1^
^3^
^4^, ^5^ ^6^
.................
_____________________ Print Screen - File # 2 _____________________
Header : None     Text : 1-6    Footer : None   1 Rec: N  Row :1    Column:1

<V>iew, <E>dit, <F3> Save, <ESC> Quit - Enter Command:  .
```

The Little Brother "define printer format" option includes a function which outputs the entire printout definition to your printer; complete with a sample record. I took the liberty of requesting this for the mailing label. This report shows the individual printer parameters (which I haven't touched on), the record field descriptions, and the sample report.

```
Printer Parms, Field Formats and Sample Screen Format for File - MAILFILE
Print Format File # 2

Physical Lines Per Page .. 6
Printed Lines Per Page ... 6
Physical Line Length ..... 60
Columns Per Record ....... 60
Left Margin Setting ...... 0
Indent on Wrap-around .... 0
Records Printed Across ... 1
Repeat Record Count ...... 1

#  Name          FLEN PLEN S      Format          Calc
-------------------------------------------------------------------------------
1  Name #1       27   27   N
2  Name #2       27   27   N
3  Address       27   27   Y
4  City          15   15   N
5  State         3    3    N
6  Zip Code      11   11   N
7  Phone         14   14   N
8  Ext           4    4    N
9  Code          6    6    N

XXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXX, XXX XXXXXXXXXXX
```

The "S" column noted above stands for "strip". This designates whether the corresponding field will have trailing spaces stripped from the printing. It would commonly be done on a first name field so that there would not be a big gap between the printing of the first and last names. The mailing label which I have defined here would be similar to that generated by MailFile. This has been an emulation of MailFile, hasn't it?

There's one last thing which I did for this data base. Back at the main menu, I selected option 11 to "set the screen". What this does is establish a particular screen automatically whenever you go into the add or update modes. Setting a default

eliminates the generation of the screen selection prompt. Since I have defined only one screen, I let Little Brother automatically designate it.

Oh yes, one more thing. I promised you a BASIC program which constructs a LB job file for adding data from any MailFile data base. I call it MF2LB/BAS. This program (which follows) reads the MailFile control file (filename/CTL) from your MailFile base and reads records in alphabetical order. In doing it this way, it ensures that only active records will be extracted from your MailFile data base. The program also includes the strange code which unpacks the telephone number field used in MailFile and keeps it a standard "N" field for LB; thus, the telephone number field can contain only digits and minus signs.

The MF2LB/BAS program is written using Model 4 BASIC. It is on DISK NOTES 7, in case you don't want to type it in. The file was saved in ASCII so it could be read by "alien" BASIC's. This dialect of BASIC should also be easily ported to MS-DOS - in case you want to convert all of your old MailFile data to the MS-DOS version of Little Brother.

```
5 REM MF2LB/BAS - Program to convert MailFile data files to Little Brother
6 REM Copyright 1987 MISOSYS, Inc., All rights reserved
7 REM Last updated: March 9, 1987
10 CLEAR 3000:DEFINT A-Z:DEFSTR S,D:CLS:GOTO 100
20 IF CVD(S6)<10000 OR S6="          " THEN P$=STRING$(8,32):RETURN
22 P$=STR$(CVD(S6)):P$=RIGHT$(P$,LEN(P$)-1):P$=LEFT$(P$,LEN(P$)-4)+"-"+RIGHT$(P$,4 )
24 IF LEN(P$)>9 THEN P$=LEFT$(P$,LEN(P$)-8)+"-"+RIGHT$(P$,8)
26 RETURN
50 LN=LEN(P$)
52 IF LEFT$(P$,1) = " " THEN P$ = "":RETURN
54 FOR L1=LN TO 1 STEP -1
56 IF(MID$(P$,L1,1)=" ") THEN GOTO 58 ELSE P$=LEFT$(P$,L1):L1=1
58 NEXT L1
60 RETURN
100 INPUT "Enter the name of your mailfile ";N$
110 OPEN"r",1,N$+"/DAT",128:FIELD 1, 27 AS S0, 27 AS S1, 27 AS S2,15 AS S3,3 AS S4, 11 AS
S5,8 AS S6,4 AS S7,6 AS S8
120 OPEN"I",2,N$+"/CNT":INPUT#2,T:REM Get # of items
130 PRINT@(1,0),"Total number of items is ";T
140 OPEN"o",3,N$+"/JOB:2"
150 PRINT#3,"2";CHR$(13);"A";
160 FOR L=1 TO T
170 INPUT#2,TA,TZ:GET 1,TA
180 PRINT@(2,0),"Record ";L;":";TA;
201 P$=S:GOSUB 50:PRINT#3,P$;CHR$(13);
202 P$=S1:GOSUB 50:PRINT#3,P$;CHR$(13);
203 P$=S2:GOSUB 50:PRINT#3,P$;CHR$(13);
204 P$=S3:GOSUB 50:PRINT#3,P$;CHR$(13);
205 P$=S4:GOSUB 50:PRINT#3,P$;CHR$(13);
206 P$=S5:GOSUB 50:PRINT#3,P$;CHR$(13);
207 GOSUB 20:GOSUB 50:PRINT#3,P$;CHR$(13);
208 P$=S7:GOSUB 50:PRINT#3,P$;CHR$(13);
209 P$=S8:GOSUB 50:PRINT#3,P$;CHR$(13);
220 PRINT#3,CHR$(27);
225 NEXT L:
226 PRINT#3,CHR$(26);
230 CLOSE:END
9999 SAVE"mf2lb/bas:2"
```

For automating your processing needs, LB can be run in an 'automatic' mode, without any operator intervention. Frequently used LB procedures (such as selecting, sorting and printing records) can be saved for future use. Entire 'Job streams' may be produced, so that LB operations may be intermixed with literally any DOS function that can be 'Batch Processed'.

LB is available for either the TRS-80 Model 4/4P under TRSDOS 6.2 (or later) or the IBM-PC/PC Compatibles under PC/MS-DOS 2.0 and operates virtually the same on either machine! As a matter of fact, data files created on one machine can be directly used on the other machine (NOTE: Separate copies of Little Brother are required and the movement of data files from one machine to another is the sole responsibility of the user). Little Brother is a "simple to use" but powerful data manager.

Hardware specifications: For the Model 4 are a minimum of two floppy disk drives and 128K of RAM (Hard disk owners need only have 64K of RAM and one floppy disk drive). For the IBM-PC are two floppy disk drives (or one hard disk and one floppy) and 128K of RAM.

Ordering Information: For the Model 4/4P => Little Brother-M4 #L-50-510
: For the IBM-PC => Little Brother-MS #L-86-510

LB Maintenance Utility

This package provides two extremely useful utility programs that can be used in conjunction with LB data files. LBMU will allow you to generate a new data base file set from existing LB data. This ability can save you countless hours in restructuring an existing data base, and provides you an excellent means to archive "old" records from your data base.

You will be allowed to create your new data base using all records in the existing file, or from a chosen few records in the file (by using a previously created LB index file). In creating the new data base file, LBMU will allocate only as many records as it needs to create the new file set. This is useful for de-allocating the space consumed by a data base file if too many records were allocated for it.

The same menu prompts That you are accustomed to in LB's Define File Format mode are used throughout when defining your new data base. Many of these prompts are answered in the exact manner as the DFF mode, so there is no need to "learn" a new program when using LBMU.

In laying out the fields to be contained in the new data base, you may use any of the fields and their associated data which are present in your existing data base. You may incorporate additional fields into the new data base. Furthermore, LBMU will allow you to lengthen or shorten any existing field, to either compress or expand your data fields as you see fit. LBMU will even perform some field type conversions for you. For example, you can convert from Numeric or Right Justified fields into Dollar/Float fields.

LBMU also allows limited data file reconstruction such as: rectifying mismatches between data and definition files; stripping out bad records from a data file while preserving as many of the good records as possible; and resurrecting an existing LB data file should its corresponding definition file become lost or unreadable.

The FIXDEL utility is included to rechain all deleted records in a LB data file should the deleted chain record chain become "broken". If you are encountering failures in re-using your deleted records, then FIXDEL is the utility program for you.

Ordering Information: For the Model 4/4P => LBMU-M4 #L-50-515
: For the IBM-PC => LBMU-MS #L-86-515